

# Algoritmos y Estructuras de Datos II

## Primer parcial

### Sábado 23 de Abril de 2016

#### Aclaraciones

- El parcial es a libro abierto.
- Cada ejercicio debe entregarse en hojas separadas.
- Incluir en cada hoja el número de orden asignado, número de hoja, apellido y nombre.
- Al entregar el parcial, completar el resto de las columnas en la planilla.
- Cada ejercicio se calificará con Promocionado, Aprobado, Regular, o Insuficiente.
- El parcial completo está aprobado si el primer ejercicio tiene al menos A, y entre los ejercicios 2 y 3 hay al menos una A. Para más detalles, ver "Información sobre la cursada" en el sitio Web.

#### Ej. 1. Especificación

El **buscaminas** es un juego cuyo objetivo es encontrar las posiciones con minas en una grilla rectangular. Llamaremos a esta **tablero**, mientras que llamaremos a sus posiciones **celdas**. Hay tres tipos de celdas: las **minadas**, que son aquellas que tienen una mina, las **pistas**, que son aquellas que son **lindantes** (en sentido horizontal, vertical o diagonal) a al menos una mina, y las **libres**, que son aquellas que no son ni minadas ni pistas. La Figura 1 muestra un ejemplo de un tablero de buscaminas.

			1	*
1	1		2	2
*	1		1	*
1	1		1	1
	1	1	1	
	1	*	1	

Figura 1: Tablero inicial, con todas sus celdas tapadas (grises).  $\star$ : mina;  $1$ : valor de una celda pista; las celdas vacías son celdas libres.

La mecánica del juego es la siguiente. Inicialmente, se dispone una cierta cantidad de minas sobre distintas posiciones del tablero. Éste se le ofrece al jugador, con todas las celdas **tapadas**<sup>1</sup>. Luego, iterativamente, el jugador elige **revelar** una a una las distintas celdas. El juego termina cuando o bien: a) el jugador logra revelar todas las celdas que no son minas (ganando el juego), o b) el jugador revela alguna celda minada (perdiendo).

Cuando el jugador decide revelar una celda  $(x,y)$ , si ésta no está libre, entonces **únicamente** se revela  $(x,y)$ . En cambio, si  $(x,y)$  es una celda libre, no sólo hay que revelar  $(x,y)$ , sino también todas sus celdas lindantes. Notar que este es un proceso recursivo; si alguna celda  $(x',y')$  lindante a  $(x,y)$  también está libre, entonces se revela dicha celda también<sup>2</sup>. Vale notar que **no** hay posibilidades de que se revele una celda minada en este proceso, aunque sí se revelarán todas las celdas pista que se encuentran al "borde" de la región formada por celdas libres (ver Figura 2).

Las **celdas pista** agregan estrategia al juego. Cada una está asociada a un número que llamaremos **la pista**, que se define como la cantidad de celdas minadas que son lindantes a la pista. Luego, a medida que el jugador va liberando las distintas pistas, puede usarlas para determinar qué posiciones tienen minas a fin de evitarlas.

Se pide modelar con un TAD el juego del buscaminas. En particular, se desea saber: **qué celdas están reveladas**; si el juego terminó y, en caso afirmativo, **con qué resultado**; y la pista asociada a cualquier celda pista ya revelada.

Además, se debe proveer una función llamada **seguroLindaAMinas** que, dado un busca minas y la posición de una celda pista  $c$  ya revelada, indique si la cantidad de celdas lindantes a la pista  $c$  que no están reveladas, coincide con el valor de la pista. En el ejemplo de la Figura 2, la función deber devolver **true** para celda inferior a la destapada, y **false** para la inferior a esta última.

	?		1	*
1	1		2	2
*	1		1	*
1	1		1	1
	1	1	1	
	1	*	1	

Figura 2: Tablero luego de destapar la celda  $\boxed{?}$ , resaltando en blanco todas las celdas destapadas por esa acción.

#### Ej. 2. Inducción estructural

Dadas las siguientes definiciones:

derechista :  $ab(\alpha) \rightarrow \text{bool}$

$d_0$      $derechista(a) = \text{nil?}(a) \vee_L (\text{nil?}(\text{izq}(a)) \wedge \text{derechista}(\text{der}(a)))$

inorder :  $ab(\alpha) \rightarrow \text{secu}(\alpha)$

$IO$      $\text{inorder}(a) \equiv \text{if nil?}(a) \text{ then } \langle \rangle \text{ else } \text{inorder}(i) \ \& \ (e \bullet \text{inorder}(d)) \ \text{fi}$

postorder :  $ab(\alpha) \rightarrow \text{secu}(\alpha)$

$PO$      $\text{postorder}(a) \equiv \text{if nil?}(a) \text{ then } \langle \rangle \text{ else } (\text{postorder}(i) \ \& \ \text{postorder}(d)) \circ e \ \text{fi}$

Demuestre por inducción estructural la siguiente propiedad sobre árboles binarios:

$$(\forall a : ab(\alpha)) ((derechista(a) \Rightarrow \text{inorder}(a) \equiv \text{reverso}(\text{postorder}(a)))$$

<sup>1</sup>En otras palabras, el jugador inicialmente no conoce el tipo de ninguna celda

<sup>2</sup>Hint: al buscar los casilleros a revelar, lleve el registro de qué casilleros ya fueron visitados para no repetirse.



En caso de utilizar lemas auxiliares, plantearlos claramente y demostrarlos. Además, se pide:

- Escribir el predicado unario. Luego escribir, completo, el **esquema de inducción** a utilizar. En el esquema, marcar **claramente** CB(s), PI(s), HI(s), TI(s) y el alcance de cada cuantificador.
- Plantear el/los caso(s) base y resolverlo(s), justificando cada paso de la demostración.
- Plantear el/los paso(s) inductivo(s) y resolverlo(s), justificando cada paso de la demostración.

### Ej. 3. Diseño

Egresados de esta facultad quieren implementar una versión local de uno de los juegos de acción más populares de los últimos tiempos. El siguiente TAD modela el comportamiento de la versión *multiplayer online* del mismo.

<b>TAD JUG es STRING, TAD NIVEL es NAT, TAD PUNTAJE es NAT</b>			
<b>TAD CARMEN ORDENA DELIVERY</b>			
<b>géneros</b>	cod		
<b>observadores básicos</b>			
jugadores	: cod	→ conj(jug)	
defNiveles	: cod	→ secu(puntaje)	
puntaje	: cod c × jug j	→ puntaje	{j ∈ jugadores(c)}
<b>generadores</b>			
sistemaOnline	: conj(jug) js × secu(puntaje) defNivel	→ cod	{long(defNivel) > 0 ∧ ordenadaEstricta(defNivel)}
peleaDelivery	: cod c × dicc(jug, dicc(jug, nat)) palos	→ cod	{ claves(palos) ⊆ jugadores(c) ∧ ((∀j: jug) def?(j, palos) ⇒ <sub>L</sub> claves(obtener(j, palos)) = <sub>obs</sub> claves(palos)) ∧ obtener(j, obtener(j, palos)) = 0 }
<b>otras operaciones</b>			
nivel	: cod c × jug j	→ nivel	{j ∈ jugadores(c)}
valores	: dicc(α, β)	→ secu(β)	
ordenadaEstricta	: secu(nat)	→ bool	
<b>axiomas</b>			
jugadores(sistemaOnline(js, n))		≡ js	
jugadores(peleaDelivery(c, palos))		≡ jugadores(c)	
defNiveles(sistemaOnline(js, n))		≡ n	
defNiveles(peleaDelivery(c, palos))		≡ defNiveles(c)	
puntaje(sistemaOnline(js, defNivel), j)		≡ 0	
puntaje(peleaDelivery(c, palos), j)		≡ puntaje(c, j) + sumar(valores(obtener(j, palos)))	
sumar(s)		≡ if vacia?(s) then 0 else prim(s) + sumar(fin(s)) fi	
valores(d)		≡ if ∅?(claves(d)) then <> else obtener(dameUno(claves(d)), d) • valores(borrar(dameUno(claves(d)), d)) fi	
ordenadaEstricta(s)		≡ if long(s) < 2 then true else prim(s) < prim(fin(s)) ∧ ordenadaEstricta(fin(s)) fi	
<b>Fin TAD</b>			

El TAD aquí presentado modela el sistema *online* para el juego *Carmen Ordena Delivery*, que trata de una señora que pide mucho delivery y da buena propina. Los jugadores interpretan a chicos y chicas de delivery en bicicleta. Cada pelea representa una vez que Carmen pide delivery. En esa pelea participan un subconjunto de jugadores. Los jugadores pueden ponerse palos en las ruedas. Cada vez que un jugador pone palos en la rueda a otros, el primero suma su puntaje. En el sistema online además hay un sistema de niveles. Cada jugador posee un nivel. El nivel al que pertenece cada jugador está definido por su puntaje. La secuencia *defNivel* que recibe el TAD en el generador base define en *defNivel[i]* el puntaje mínimo para alcanzar el nivel *i*. Esta secuencia debe estar ordenada de forma creciente y estricta para que siempre esté definido a qué nivel pertenece un jugador.

Para representar el TAD CARMEN ORDENA DELIVERY se decidió utilizar la siguiente estructura:

cod se representa con estr, donde
estr es tupla ( jugadores: conj(jug),
puntajeANivel: dicc(puntaje, nivel),
nivelJugador: dicc(jug, nivel),
palosTot: dicc(jug, dicc(jug, nat))

donde *jugadores* es el conjunto de jugadores del sistema online, *puntajeANivel* nos dice para cada puntaje que sube el nivel del jugador a qué nivel sube el mismo, *nivelJugador* es el nivel para cada jugador y *palosTot* es un diccionario donde para cada jugador tiene un diccionario diciendo para cada otro jugador, cuantos palos le puso en las ruedas en total.

- Escribir en castellano el invariante de representación.
- Escribir formalmente el invariante de representación.
- Escribir formalmente la función de abstracción.



1) TAD Celda es Tupla (nat, nat)

TAD Pista es Nat

// 0 significa que no hay pista

TAD Buscaminas

generos bm

exporta bm, generadores, observadores, otras operaciones

usa Nat, Conj(x), Bool, Pista, Celda

igualdad obs  $(\forall b, b' : bm) \{ b =_{obs} b' \Leftrightarrow (alto(b) = alto(b') \wedge ancho(b) = ancho(b') \wedge minas(b) = minas(b') \wedge revelado(b) = revelado(b')) \}$

observadores básicos

alto:  $bm \rightarrow nat$

ancho:  $bm \rightarrow nat$

minas:  $bm \rightarrow conj(celda)$

revelado:  $bm \rightarrow conj(celda)$

generadores

nuevo:  $nat \times nat \times conj(celda) \times minas \rightarrow bm$

$\{ \neg 0?(x) \wedge \neg 0?(y) \wedge (\forall c : celda, c \in minas) (\pi_1(c) < x \wedge \pi_2(c) < y) \}$

revelar:  $bm \ b \times celda \ c \rightarrow bm$

$\{ \pi_1(c) < ancho(b) \wedge \pi_2(c) < alto(b) \wedge \neg (c \in revelado(b)) \wedge \neg terminado(b) \}$

otras ops

termino?:  $bm \ b \rightarrow bool$

gano?:  $bm \ b \rightarrow bool$

seguro Linda A Mina:  $bm \ b \times celda \ c \rightarrow bool$

pista:  $bm \ b \times celda \ c \rightarrow pista$

$\{ terminado?(b) \}$

$\{ celda Valida(b, c) \wedge (pista(b, c) > 0 \wedge (c \in revelado(b, c))) \}$   
 $\{ celda Valida(b, c) \}$



1	2	3
P	P	P



expandir(b, todo, done)  $\equiv$  if  $\emptyset^?(todo)$  then  
done

else

if  $\text{pistaN}(b, \text{dameUno}(todo)) > 0$   $\wedge$   $\text{dameUno}(todo) \in \text{muras}(b)$  then  
expandir(b, sinUno(todo), Ag(dameUno(todo), done))

else

expandir(b, sinUno(todo)  $\cup$  (lindantes(b, dameUno(todo)) - done),  
Ag(dameUno(todo), done))

fi

fi

~~lindantes(b, c)  $\equiv$  ~~lindantes(b, c)  $\equiv$  Ag(~~sinUno(todo)~~~~~~

(if  $\pi_1(c) \equiv 0$  then  $\emptyset$  else

(if  $\pi_2(c) = 0$  then  $\emptyset$  else Ag( $\langle \pi_1(c) - 1, \pi_2(c) - 1 \rangle, \emptyset$ ) fi)  $\cup$

(if  $\pi_2(c) + 1 = \text{alto}(b)$  then  $\emptyset$  else Ag( $\langle \pi_1(c) - 1, \pi_2(c) + 1 \rangle, \emptyset$ ) fi)  $\cup$

Ag( $\langle \pi_1(c) - 1, \pi_2(c) \rangle, \emptyset$ )

fi)  $\cup$

(if  $\pi_1(c) + 1 = \text{ancho}(b)$  then  $\emptyset$  else

(if  $\pi_2(c) = 0$  then  $\emptyset$  else Ag( $\langle \pi_1(c) + 1, \pi_2(c) - 1 \rangle, \emptyset$ ) fi)  $\cup$

(if  $\pi_2(c) + 1 = \text{alto}(b)$  then  $\emptyset$  else Ag( $\langle \pi_1(c) + 1, \pi_2(c) + 1 \rangle, \emptyset$ ) fi)  $\cup$

Ag( $\langle \pi_1(c) + 1, \pi_2(c) \rangle, \emptyset$ )

)  $\cup$

(if  $\pi_2(c) = 0$  then  $\emptyset$  else Ag( $\langle \pi_1(c), \pi_2(c) - 1 \rangle, \emptyset$ ) fi)  $\cup$

(if  $\pi_2(c) + 1 = \text{alto}(b)$  then  $\emptyset$  else Ag( $\langle \pi_1(c), \pi_2(c) + 1 \rangle, \emptyset$ ) fi)



$expandir(b, todo, done) \equiv$  if  $\emptyset?(todo)$  then

done

else

if  $pistaN(b, dameUno(todo)) > 0$  &  $dameUno(todo) \in minas(b)$  then  
 $expandir(b, sinUno(todo), Ag(dameUno(todo), done))$

else

$expandir(b, sinUno(todo) \cup (lindantes(b, dameUno(todo)) - done),$   
 $Ag(dameUno(todo), done))$

fi

fi

$lindantes(b, c) \equiv$   ~~$lindantes(b, c) \equiv Ag(\{ \pi_1(c) \pm 1, \pi_2(c) \pm 1 \}, \emptyset) fi$~~

(if  $\pi_1(c) \pm 1$  then  $\emptyset$  else

(if  $\pi_2(c) = 0$  then  $\emptyset$  else  $Ag(\langle \pi_1(c) - 1, \pi_2(c) - 1 \rangle, \emptyset) fi) \cup$

(if  $\pi_2(c) + 1 = alto(b)$  then  $\emptyset$  else  $Ag(\langle \pi_1(c) - 1, \pi_2(c) + 1 \rangle, \emptyset) fi) \cup$

$Ag(\langle \pi_1(c) - 1, \pi_2(c) \rangle, \emptyset)$

fi)  $\cup$

(if  $\pi_1(c) + 1 = ancho(b)$  then  $\emptyset$  else

(if  $\pi_2(c) = 0$  then  $\emptyset$  else  $Ag(\langle \pi_1(c) + 1, \pi_2(c) - 1 \rangle, \emptyset) fi) \cup$

(if  $\pi_2(c) + 1 = alto(b)$  then  $\emptyset$  else  $Ag(\langle \pi_1(c) + 1, \pi_2(c) + 1 \rangle, \emptyset) fi) \cup$

$Ag(\langle \pi_1(c) + 1, \pi_2(c) \rangle, \emptyset)$

$\cup$

(if  $\pi_2(c) = 0$  then  $\emptyset$  else  $Ag(\langle \pi_1(c), \pi_2(c) - 1 \rangle, \emptyset) fi) \cup$

(if  $\pi_2(c) + 1 = alto(b)$  then  $\emptyset$  else  $Ag(\langle \pi_1(c), \pi_2(c) + 1 \rangle, \emptyset) fi)$



$$ce(d, U_0(b, c)) = \pi_1(c) < ancho(b) \wedge \pi_2(c) < alto(b)$$

$$\#minas(b, cs) = \text{if } \emptyset^9(cs) \text{ then } 0 \text{ else } \beta(d, U_0(cs) \in minas(b)) \\ + \#minas(b, sin U_0(cs))$$



2. a)

Predicado unario:

$$P(a) \equiv (\text{derechista}(a) \Rightarrow \text{inorder}(a) \equiv \text{reverso}(\text{posorder}(a)))$$

Esquema de inducción

$$\frac{\underbrace{P(\text{nil})}_{\text{CB}} \wedge (\forall a: ab(x)) \left( \underbrace{P(a)}_{\text{HI}} \Rightarrow (\forall b: ab(x)) (\forall e: x) \underbrace{P(\text{bin}(b, e, a))}_{\text{TI}} \right)}{\text{PI}}$$

b) Caso base:

$$P(\text{nil}) \stackrel{\text{PV}}{\equiv} (\text{derechista}(\text{nil}) \Rightarrow \text{inorder}(\text{nil}) \equiv \text{reverso}(\text{posorder}(\text{nil})))$$

Intentamos probar que el consecuente siempre es verdadero

$$(\text{inorder}(\text{nil}) \equiv \text{reverso}(\text{posorder}(\text{nil})))$$

$$\stackrel{\text{IOPD}}{\equiv} (\text{if nil?}(\text{nil}) \text{ then } \langle \rangle \text{ else } \text{inorder}(a) \& (e \cdot \text{inorder}(a)) \text{ fi} \equiv \text{reverso}(\text{if nil?}(\text{nil}) \text{ then } \langle \rangle \text{ else } (\text{postorder}(a) \& \text{postorder}(a)) \text{ oie fi}))$$

Como  $\text{nil?}(\text{nil}) \equiv \text{true}$ , reducimos ambos ifs

$$\equiv (\langle \rangle \equiv \text{reverso}(\langle \rangle))$$

$$\stackrel{\text{PI}}{\equiv} (\langle \rangle \equiv \langle \rangle) \quad \square$$

~~Como~~ Como el consecuente es siempre verdadero, la implicación es siempre verdadera y  $P(\text{nil}) \equiv \text{true}$ 

c) Paso inductivo

$$\text{Sean } a, b: ab(x), e: x$$

$$\text{HI: } P(a)$$

$$\text{TI: } P(\text{bin}(b, e, a))$$

$$\text{TI: } P(\text{bin}(b, e, a))$$

Podemos separar dos casos,  $\neg \text{nil}(b)$  y  $\text{nil}(b)$



Caso 1,  $nil(b)$ :

Nos basta con demostrar que el antecedente de la implicación en el PV es siempre falso.

derechista(bin(b, e, a))

$\equiv nil?(bin(b, e, a) \vee_L (nil?(izq(bin(b, e, a)) \wedge derecha(der(bin(b, e, a))))$

$\equiv nil? \wedge izq \wedge der \quad false \vee_L (nil? \wedge izq \quad derecha \quad der)$

Hipótesis caso 1  $\equiv false \vee_L (false \wedge derecha(a))$

$\wedge \equiv false \vee_L false$

$\vee_L \equiv false$

Como el antecedente es falso, la implicación es siempre verdadera  $\Rightarrow P(bin(b, e, a)) = true$  (en este caso)

Caso 2,  $nil(b)$ :

~~Veremos que en este caso el consecuente es siempre verdadero si el antecedente es verdadero, y si es falso la implicación vale trivialmente.~~

Para abreviar, llamaremos  $der$  a  $derechista$ ,  $in$  a  $inorden$ ,  $pos$  a  $postorden$  y  $rev$  a  $reverso$ .

Hipótesis caso 2:  $derechista(bin(b, e, a))$

~~$(in(bin(b, e, a)) \equiv rev(pos(bin(b, e, a))))$~~

$\equiv (if \quad nil?(bin(b, e, a)) \text{ then } \langle \rangle \text{ else } in(b) \& (e \circ in(a))) fi$

$\equiv rev(if \quad nil?(bin(b, e, a)) \text{ then } \langle \rangle \text{ else } (pos(b) \& pos(a)) \circ e \text{ fi})$

Como  $nil?(bin(b, e, a)) = false$ , reducimos ambos if's

$\equiv (in(b) \& (e \circ in(a))) \equiv rev((pos(b) \& pos(a)) \circ e)$

Lema 1:  $(\forall b: ab(x), nil?(b)) (in(b) \equiv \langle \rangle)$

Lema 2:  $(\forall b: ab(x), nil?(b)) (pos(b) \equiv \langle \rangle)$

Como  $nil?(b)$  por hipótesis del caso 2,

$\equiv (Lema 1 \wedge Lema 2) \{ (\langle \rangle \& (e \circ in(a))) \equiv rev((\langle \rangle \& pos(a)) \circ e) \}$



4/5

$$\&_1 \equiv (e \circ \text{in}(a) \equiv \text{rev}(\text{pos}(a) \circ e))$$

$$\text{Lema 3: } (\forall e: \alpha) (\forall a, b: \text{ab}(\alpha), \text{nil}?(b) \wedge \text{der}(\text{bin}(b, e, a))) \text{der}(a)$$

$$\text{Lema 4: } (\forall s: \text{seco}(\alpha)) (\forall e: \alpha) (e \circ \text{rev}(s) \equiv \text{rev}(s \circ e))$$

Podemos usar el lema 3 por  $\text{nil}?(b)$  por hipótesis caso 2 y  $\text{der}(\text{bin}(b, e, a))$  por hipótesis caso 2

Ahora, por lema 3,  $\text{der}(a)$  y entonces por HI,  $\text{in}(a) \equiv \text{rev}(\text{pos}(a))$

$$\stackrel{\text{HI}}{\equiv} (e \circ \text{rev}(\text{pos}(a)) \equiv \text{rev}(\text{pos}(a) \circ e))$$

$$\stackrel{\text{Lema 4}}{\equiv} (e \circ \text{rev}(\text{pos}(a)) \equiv (e \circ \text{rev}(\text{pos}(a)))) \quad \square$$

Como probamos todos los casos, vale el PI

No falta probar los lemas:

$$\text{Lema 1: } (\forall b: \text{ab}(\alpha), \text{nil}?(b)) (\text{in}(b) \equiv \langle \rangle)$$

Sea  $b$ , tq.  $\text{nil}?(b)$   $\stackrel{\text{true}}{\text{true}}$

$$\text{in}(b) \stackrel{\text{IO}}{\equiv} \text{if } \text{nil}?(b) \text{ then } \langle \rangle \text{ else } \text{in}(i) \& (e \circ \text{in}(d)) \text{ fi}$$

$$\stackrel{\text{if}}{\equiv} \langle \rangle \quad \square$$

$$\text{Lema 2: } (\forall b: \text{ab}(\alpha), \text{nil}?(b)) (\text{pos}(b) \equiv \langle \rangle)$$

Sea  $b$ , tq.  $\text{nil}?(b)$   $\stackrel{\text{true}}{\text{true}}$

$$\text{pos}(b) \stackrel{\text{IO}}{\equiv} \text{if } \text{nil}?(b) \text{ then } \langle \rangle \text{ else } (\text{pos}(i) \& \text{pos}(d)) \circ e \text{ fi}$$

$$\stackrel{\text{if}}{\equiv} \langle \rangle \quad \square$$

$$\text{Lema 3: } (\forall e: \alpha) (\forall a, b: \text{ab}(\alpha), \text{nil}?(b) \wedge \text{der}(\text{bin}(b, e, a))) \text{der}(a)$$

Sean  $e: \alpha$ ,  $a, b: \text{ab}(\alpha)$  tq.  $\text{nil}?(b) \wedge \text{der}(\text{bin}(b, e, a))$

$\stackrel{\text{H4}}{\text{true}} \equiv \text{der}(\text{bin}(b, e, a))$

$$\stackrel{\text{do}}{\equiv} \text{nil}?(b) \wedge \text{izq}(\text{der}(\text{bin}(b, e, a))) \vee \text{nil}?(b) \wedge \text{der}(\text{der}(\text{bin}(b, e, a)))$$

$$\stackrel{\text{nil?} \wedge \text{izq} \wedge \text{der}}{\equiv} \begin{matrix} \text{false} & \vee & \text{true} \\ \text{false} & \vee & \text{true} \end{matrix} \text{nil}?(b) \wedge \text{der}(a)$$

$$\stackrel{\vee}{\equiv} \text{nil}?(b) \wedge \text{der}(a)$$



Por hipótesis,  $nil?(b) \in true$

$$\stackrel{\text{Hipótesis}}{=} true \wedge der(a) \stackrel{\wedge}{=} der(d)$$

$$\Rightarrow true \equiv der(d) \quad \square$$

Lema 4:  $(\forall s: secu(\alpha)) (\forall e: \alpha) (e \circ rev(s) \equiv rev(s \circ e))$

Probamos inductivamente

$$PU: P(s) = (\forall e: \alpha) (e \circ rev(s) \equiv rev(s \circ e))$$

$$EI: P(<>) \wedge (\forall s: secu(\alpha)) (P(s) \Rightarrow (\forall e: \alpha) P(e \circ s))$$

Caso base:

$$\begin{aligned} P(<>) &\equiv (e \circ rev(<>) \equiv rev(<> \circ e)) \\ &\stackrel{rev_1, o_1}{=} (e \circ <> \equiv rev(e \circ <>)) \\ &\stackrel{rev_2}{=} (e \circ <> \equiv rev(<>) \circ e) \\ &\stackrel{rev_1}{=} (e \circ <> \equiv <> \circ e) \\ &\stackrel{o_1}{=} (e \circ <> \equiv e \circ <>) \quad \square \end{aligned}$$

Paso inductivo

$$HI: P(s)$$

$$P(e' \circ s) \equiv (e' \circ rev(e' \circ s) \equiv rev((e' \circ s) \circ e))$$

$$\stackrel{rev_2}{=} (e' \circ (rev(s) \circ e') \equiv rev((e' \circ s) \circ e))$$

$$\stackrel{o_2}{=} ((e' \circ rev(s)) \circ e' \equiv rev(e' \circ (s \circ e)))$$

$$\stackrel{rev_2}{=} ((e' \circ rev(s)) \circ e' \equiv rev((e' \circ s) \circ e))$$

$$\stackrel{HI}{=} (rev(s \circ e) \circ e' \equiv rev(s \circ e) \circ e') \quad \square$$



- a) - El puntaje de cada nivel en puntajeANivel ~~es~~ es <sup>0</sup>es creciente, y hay niveles sin huecos, y no se repiten
- Las claves de nivelJugador son todos los jugadores
  - Las claves de palosTotales son todos los jugadores
  - Los valores de palosTot tienen todos los jugadores
  - Un jugador no tiene palos así mismo en palosTot
  - Los niveles en nivelJugador concuerdan con palosTot y puntajeANivel

b) Rep:  $\text{estr} \rightarrow \text{bool}$

( $\forall e: \text{estr}$ )

Rep(e)  $\equiv$  ~~no hay claves (e.puntajeANivel)~~

~~( $\forall p: \text{puntaje}, p \in \text{claves}(e.puntajeANivel)$ )~~

( $\text{obtener}(p, e.puntajeANivel) > 0 \Rightarrow$

( $\exists p': \text{puntaje}, p \in \text{claves}(e.puntajeANivel)$ )

( $p' < p \wedge (\text{obtener}(p, e.puntajeANivel) = \text{obtener}(p', e.puntajeANivel) + 1)$ )

~~)~~

$\text{claves}(e.nivelJugador) = e.jugadores \wedge$

$\text{claves}(e.palosTot) = e.jugadores \wedge$

( $\forall j: \text{jug}, j \in e.jugadores$ )

( $\text{claves}(\text{obtener}(j, e.palosTot)) = e.jugadores \wedge$

$\text{obtener}(j, \text{obtener}(j, e.palosTot)) = 0$ .)

$\wedge \text{def?}(0, e.puntajeANivel) \wedge (\text{obtener}(0, e.puntajeANivel) = 0)$

$\wedge (\#(\text{claves}(e.puntajeANivel)) > 1)$

$\wedge (\forall j: \text{jug}, j \in e.jugadores)$

~~( $\forall p: \text{puntaje}, p \neq \text{obtener}(j, e.palosTot) \wedge p \in \text{claves}(e.puntajeANivel)$ )~~

~~$p \in \text{obtener}(j, e.palosTot)$~~

$\wedge$  \* sigue después de Abs



c)  $Abs: \hat{estr} \rightarrow cod$

$\{Rep(e)\}$

$(\forall e: \hat{estr}, Rep(e)) (\forall c: cod)$

$Abs(e) =_obs c \mid jugadores(c) = e.jugadores \wedge$

$(\forall j: jug, j \in jugadores(c))$

$(punteaje(c, j) = \text{sumar}(\text{obtener}(j, e.palosTot)))$

$\wedge (\text{long}(c.defNiveles) = \#(claves(e.puntANiv)) - 1)$

$\wedge (\forall p \in punteaje, p \in claves(e.puntANiv) \wedge \neq 0?(p))$

$\text{est}^? (p, c.defNiveles)$

\* (sigue Rep)

~~$\wedge (\forall j: jug, j \in e.jugadores)$~~

$\wedge (\forall j: jug, j \in e.jugadores)$

$(\exists p: punteaje, p \in claves(e.puntANiv))$

$(p \leq \text{sumar}(\text{obtener}(j, e.palosTot))) \wedge$

$\wedge$

$(\forall p: punteaje, p \in claves(e.puntANiv))$

$(p > \text{sumar}(\text{obtener}(j, e.palosTot))) \Rightarrow$

$\text{obtener}(j, e.nivelJug) < \text{obtener}(p, e.puntANiv)$

)